K. J. Somaiya College of Engineering, Mumbai
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Batch: A3	**Roll No.: 16010121045**
Experiment / assignment / tutorial No. 4	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Implementation of Stack applications.

Objective: To implement applications of stack

Expected Outcome of Experiment:

CO	Outcome
1	Explain the different data structures used in problem solving

Books/ Journals/ Websites referred:

1. *Fundamentals Of Data Structures In C* – Ellis Horowitz, Satraj Sahni, Susan Anderson-Fred
2. *An Introduction to data structures with applications* – Jean Paul Tremblay, Paul G. Sorenson
3. *Data Structures A Pseudo Approach with C* – Richard F. Gilberg & Behrouz A. Forouzan
4. *https://www.cprogramming.com/tutorial/computersciencetheory/stack.html*
5. *https://www.geeksforgeeks.org/stack-data-structure-introduction-program/*
6. *https://www.thecrazyprogrammer.com/2013/12/c-program-for-array-representation-of-stack-push-pop-display.html*

Assigned Stack application: Static, Undo-Redo operations

Algorithm:

- Start
- Initialize two stacks, say Stack and Backup-stack.
- Traverse the array of strings, Q, and perform the following operations:
- If Add data is selected then, push the character to Undo stack
- If Undo data is selected then, pop the top element from main stack and push it to backup stack.
- If Redo data is selected then, pop the top element of backup stack and push it into the main stack.
- If display data is selected then, print all the elements of the main stack.

Example:

- Perform Write P in the data. Therefore, the data contains only "P".
- Perform Write A on the data. Therefore, the data contains "PA".
- Perform Write R on the data. Therefore, the data contains "PAR".
- Perform Undo on the data. Therefore, the data contains "PA".
- Print the contents of the data, i.e. "PA"
- Perform Redo on the data. Therefore, the data contains "PAR".
- Print the contents of the data, i.e. "PAR"

Source code:

```
#include <stdio.h>
void push(char *arr, char ele, int n, int *top)
{
    if (*top >= n - 1)
        printf("The Stack is Full\n");
    else
    {
        *top = *top + 1;
        arr[*top] = ele;
    }
}

char pop(char *arr, int *top)
{
    if (*top != -1)
    {
        int temp = arr[*top];
        arr[*top] = 0;
        *top = *top - 1;
        return (temp);
    }
    printf("The Stack is Empty\n");
    return ('@');
}

char peek(char *arr, int *top)
{
    return (arr[*top]);
```

```
}

void printStack(char *arr, int *top)
{
    for (int i = *top; i > -1; i--)
        printf("%c\n", arr[i]);
}

int main()
{
    int n;
    printf("Enter Opperation size: ");
    scanf("%d", &n);
    char stack[n], bstack[n];
    int top = -1, btop = -1;
    int j = -1;
    while (j != 5)
    {
        printf("\n\n(1) Add Data\n");
        printf("(2) Undo\n");
        printf("(3) Redo\n");
        printf("(4) View Data\n");
        printf("(5) Exit\n");
        printf("Select an option: ");
        scanf("%d", &j);
        if (j == 1)
        {
            char ele;
            printf("\nEnter Data: ");
```

```
        scanf(" %c", &ele);
        push(stack, ele, n, &top);
    }
    else if (j == 2)
    {
        char temp = pop(stack, &top);
        if (temp != '@')
        {
            push(bstack, temp, n, &btop);
            printf("Element %c is removed\n", temp);
            if (peek(stack, &top) == '@')
                printf("There is no next data!\n");
            else
                    printf("Next data is %c", peek(stack, &top));
        }
    }
    else if (j == 3)
    {
        char temp = pop(bstack, &btop);
        if(temp !='@'){
        push(stack, temp, n, &top);
        printf("Element %c is added back!\n", temp);
        }
    }
    else if (j == 4)
        printStack(stack, &top);
    else if (j == 5)
        break;
    else
```

```
        printf("Please Enter correct option\n");
    }
    return 0;
}
```

Output Screenshots:

```
Enter Opperation size: 5


(1) Add Data
(2) Undo
(3) Redo
(4) View Data
(5) Exit
Select an option: 1

Enter Data: P


(1) Add Data
(2) Undo
(3) Redo
(4) View Data
(5) Exit
Select an option: 1

Enter Data: A


(1) Add Data
(2) Undo
(3) Redo
(4) View Data
(5) Exit
Select an option: 1

Enter Data: R


(1) Add Data
(2) Undo
(3) Redo
(4) View Data
(5) Exit
Select an option: 4
```

```
Select an option: 4
R
A
P


(1) Add Data
(2) Undo
(3) Redo
(4) View Data
(5) Exit
Select an option: 2
Element R is removed
Next data is A

(1) Add Data
(2) Undo
(3) Redo
(4) View Data
(5) Exit
Select an option: 2
Element A is removed
Next data is P

(1) Add Data
(2) Undo
(3) Redo
(4) View Data
(5) Exit
Select an option: 3
Element A is added back!


(1) Add Data
(2) Undo
(3) Redo
(4) View Data
(5) Exit
Select an option: 3
Element R is added back!
```

```
(1) Add Data
(2) Undo
(3) Redo
(4) View Data
(5) Exit
Select an option: 4
R
A
P


(1) Add Data
(2) Undo
(3) Redo
(4) View Data
(5) Exit
Select an option: 5
pargat@Router Exp4 %
```

Conclusion:
Successfully implemented Undo-Redo operation implementation using static stack.